

# **Utveckling av en personalschemalägnings- webbapplikation med ramverket CodeIgniter**

Jens Klemets

EXAMENSARBETE	
Arcada	
Utbildningsprogram:	Informations- och medieteknik
Identifikationsnummer:	4191
Författare:	Jens Klemets
Arbetets namn:	Utveckling av en personalschemalägnings- webbapplikation med ramverket CodeIgniter
Handledare (Arcada):	Johnny Biström
Uppdragsgivare:	Oy Cubelux Ab
<p>Sammandrag:</p> <p>Examensarbetet behandlar hur man använt PHP ramverket CodeIgniter för att utveckla en personalschemalägningsapplikation. Idéen till applikationen kommer från Patrik Nylund. Syftet med arbetet är att visa hur man gått tillväga för att skapa applikationen och ge den kommande användaren en bild av hur man ska använda applikationen. Applikationen kommer också att bli en produkt i företaget Oy Cubelux Ab:s applikations utbud. Man önskar underlätta små och medelstora företags personalschemaläggning med applikationen.</p> <p>Författaren har utgått från att läsaren har en insikt i webbutveckling och objektorienterad programmering. Det första kapitlet är en inledning till examensarbetet där man ger läsare en insikt om vad som kommer att behandlas och vad man tänkt sig att applikationen skall bli. De tekniker man har använt för att kunna förverkliga applikationen beskrivs teoretiskt i det andra kapitlet. Man har beskrivit implementationen av applikationen i det tredje kapitlet. Här får läsaren ta del av databasstrukturen, gränssnittets utveckling och hur informationen skall hanteras. Det fjärde kapitlet visar läsaren vilka resultat man fått. Här kan läsaren ta del av kodexempel från applikationen och se hur man skall använda applikationen. Det sista kapitlet är författarens egna tankar om utveckling med ett PHP ramverk.</p>	
Nyckelord:	MVC, PHP, jQuery, CodeIgniter, webbutveckling, webbapplikation, Cubelux
Sidantal:	42
Språk:	Svenska
Datum för godkännande:	

DEGREE THESIS	
Arcada	
Degree Programme:	Information and Media Technology
Identification number:	4191
Author:	Jens Klemets
Title:	Development of a staff scheduling web application with CodeIgniter Framework
Supervisor (Arcada):	Johnny Biström
Commissioned by:	Oy Cubelux Ab
<p><b>Abstract:</b></p> <p>The thesis describes how to use PHP framework CodeIgniter to develop a staff scheduling application. The idea for the application comes from Patrik Nylund. The purpose of this work is to show how I have proceeded to create the application and give the next user a picture of how to use the application. The application will also be a product of the company Oy Cubelux Ab's application range. It wishes to facilitate small and medium-sized enterprises with staff scheduling application.</p> <p>I assume that the reader has an understanding of web development and object-oriented programming when they are reading the thesis. The first chapter is an introduction to the thesis that give readers an insight of what will be discussed and what you thought that the application should be. The techniques I have used to realize the application described theoretically in the second chapter. I have described the implementation of the application in the third chapter. Here the reader can grasp the database structure, interface development, and how information should be handled. The fourth chapter shows the reader what results I got. Here the reader can take advantage of code samples from the application and see how to use the application. The last chapter is my own thoughts on the development of a PHP framework.</p>	
Keywords:	MVC, PHP, jQuery, CodeIgniter, web development, web application, Cubelux
Number of pages:	42
Language:	Swedish
Date of acceptance:	



# INNEHÅLL

<b>1</b>	<b>Inledning.....</b>	<b>10</b>
1.1	Bakgrund.....	10
1.2	Syfte och målgrupp .....	10
1.3	Avgränsning .....	11
1.4	Programmeringsspråk och verktyg .....	11
<b>2</b>	<b>Metod .....</b>	<b>13</b>
2.1	Databashanteraren .....	13
2.2	Webbapplikationsramverk.....	14
2.3	Model-View-Controller.....	15
2.4	CodeIgniter.....	16
2.4.1	Kort översikt av ramverket .....	16
2.4.2	MVC för att organisera applikationen .....	17
2.4.3	Autentiseringsbiblioteket Ion Auth.....	17
2.4.4	CodeIgniters mål.....	17
2.5	JavaScript .....	18
2.5.1	jQuery.....	18
2.5.2	Modernizr .....	19
<b>3</b>	<b>Implementation .....</b>	<b>21</b>
3.1	Databasstrukturen.....	21
3.2	Gränssnittet.....	24
3.3	Informationshanteringen .....	25
3.3.1	Hanteringen av avdelningar .....	25
3.3.2	Hantering av användare.....	26
3.3.3	Hanteringen av skiften .....	28
3.3.4	Hanteringen av planerad arbetstid.....	29
3.3.5	Hanteringen av skiften för anställda.....	29
<b>4</b>	<b>Resultat.....</b>	<b>30</b>
4.1	Template .....	30
4.2	Inloggningsportalen.....	31
4.3	Administrator vyn .....	33
4.4	Anställdas vy .....	39
<b>5</b>	<b>Diskussion.....</b>	<b>41</b>
	<b>Källor .....</b>	<b>42</b>



## Figurer

Figur 1. CodeIgniters flödesschema (EllisLab, Inc 2012).....	17
Figur 2. Databastabellen users.....	21
Figur 3. Databastabellen department.....	22
Figur 4. Databastabellen department_users.....	22
Figur 5. Databastabellen shifts.....	22
Figur 6. Databastabellen plannedtime.....	23
Figur 7. Databastabellen available.....	23
Figur 8. Databastabellen days and time.....	23
Figur 9. Kodexempel på hur man kan visa en vy i CodeIgniter.....	30
Figur 10. Kodexempel på hur man kan visa en text sträng ur en nyckel.....	30
Figur 11. Kodexempel på hur man skapat en bas mall.....	31
Figur 12. Ett exempel ur autentiserings kontrollern.....	31
Figur 13. Inloggningsportalens vy.....	32
Figur 14. Kodexempel från inloggningsportalens kontroller.....	33
Figur 15. Administrator vyns Dashboard.....	33
Figur 16. Avdelnings vyn.....	34
Figur 17. Vyn för att lägga till en avdelning.....	34
Figur 18. Vyn för att editera en avdelning.....	34
Figur 19. Vyn som visar alla anställda.....	35
Figur 20. Vyn för att lägga till en ny anställd.....	35
Figur 21. Vy för att lägga till en anställd i en avdelning.....	36
Figur 22. Vyn för att editera anställd.....	36
Figur 23. Vyn som presenterar alla skiften i applikationen.....	37
Figur 24. Vy för att ge en anställd ett skifte.....	37
Figur 25. Vy för att skapa ett tillfälligt skifte.....	38
Figur 26. Vy för att skapa ett tillfälligt skifte med kalender.....	38
Figur 27. Vyn för all planerad tid.....	39
Figur 28. Vy med bokade extra skiften.....	39
Figur 29. Vyn för att boka ett extra skifte.....	40
Figur 30. Vyn för att editera anställds information.....	40





## FÖRORD

Jag har alltid varit fascinerad över webbsidor och deras funktioner. Valet att göra en applikation som examensarbete var inte svårt. Genast när min vän Patrik Nylund presenterade sin idé om att skapa en webbapplikation för att underlätta personal schemaläggningen såg jag min möjlighet att lära mig något nytt genom att skapa någonting många kan ha användning för.

Möjligheter att lära sig det man vill och influenser från att ha utvecklat många olika typs webbsidor åt kunder med olika tekniker, har varit starkt närvarande när jag gjort denna applikation. Efter att ha gjort webbsidor i skolan och åt egna klienter har jag fått mycket erfarenhet av webbt tekniker tillgängliga.

Många har ifrågasatt mitt val av att skapa en webbapplikation, men efter att ha uppnått mitt mål med applikationen känns det som jag gjorde rätt val. Jag hoppas jag kan hjälpa så många företag jag bara kan med effektivisering av personalschemaläggning.

Jag vill tacka alla er som har varit med och hjälpt mig med förberedelser, agerat bollplank och stött mig under processen. Speciellt stort tack vill jag tillägna min familj för deras stöd och Satu för hennes stora tålamod och intresse.

*Quod scripsi scripsi* – Pontius Pilatus

# 1 INLEDNING

Examensarbete är utfört för Oy Cubelux Ab som är en ny aktör på marknaden och vill snabbt komma igång med att effektivera vardagen för deras kunder. Cubelux hade också behov av större kunnskap inom applikationsutvecklingsområdet. Detta gjorde valet av att göra en webbaserad applikation för framtida användare en klar fördel på marknaden för Cubelux. Cubelux anser att användarna snabbt skall komma åt information och ha möjligheten att använda webbapplikationen på flera olika plattformar.

## 1.1 Bakgrund

Företaget Oy Cubelux Ab är ett startup företag startat 2011. Deras affärsområden är programvaruutveckling, webbutveckling, design och branding.

De fick idén om att utveckla en webbapplikation för personalschemaläggning i små och medelstora företag genom en av deras ägare, som länge har sett att en kommersiellt tillgänglig och billig personalschemalägningswebbapplikation kunde vara en bra investering. Applikationen beställdes av Cubelux och jag valdes som utvecklare för webbapplikationen.

Cubelux visste att det fanns många färdiga lösningar, men man ville ändå försöka sig på att skapa en webbapplikation för den finländska marknaden av ett finländskt bolag. Patrik Nylund hade tagit reda på ifall det fanns en liknande produkt från Finland, men hittade inte en som skulle uppfylla de krav han hade ställt.

## 1.2 Syfte och målgrupp

Webbapplikationen ska bli ett av Cubelux utbud av webbapplikationer som kan komma att underlätta många personalrekryterares arbete med att söka tillgänglig personal. Examensarbetet kommer också att ge mig mycket bättre insikt i PHP ramverk och öka min kunskap inom webbapplikationsutveckling.

Målen är att ha en färdig alfa version av applikationen för företag att testa och att man ska kunna komma åt applikationen online för att kunna logga in på webbapplikationen. Webbapplikationen kommer främst att utvecklas för planering av personalbehov i små och medelstora företag. Man har genom att komma ihåg tidigare arbetsplatsers arbetstider, avdelningar och personalstyrka sett ett större behov av flexibilitet och tillgänglighet av en applikation för arbetstagare.

### **1.3 Avgränsning**

Webbapplikationer byggda med ett PHP- ramverk behöver en databas för att man skall dynamiskt kunna presentera innehåll i webbapplikationen. Den använda relationsdatabasen kommer därför att presenteras för läsaren.

Metodiken för att ta fram det grafiska gränssnittet kommer att gås igenom. För att uppnå en robust och användarvänlig webbapplikation kommer jag att använda ett populärt JavaScript bibliotek (jQuery) med stöd för alla moderna webbläsare. jQuery kommer också att spela en roll i användningen av applikationen och datahanteringen i applikationen.

Beskrivningar över den framarbetade kravspecifikationen och det planerade användargränssnittet kommer att presenteras för att läsaren skall kunna följa programmeringsprocessen. Teori om hur man programmerar i ett särskilt språk (syntax) kommer inte att tas med då läsaren förutsätts ha viss insikt i detta. Jag kommer som avslutning diskutera framtidsutsikten för webbapplikationen, utökning av andra applikationer och möjliga tillägg för applikationen.

### **1.4 Programmeringsspråk och verktyg**

En webbaserad applikation som skall klara av en stor användarskara behöver en bra databashanterare. MySQL är vida använt i många webbapplikationer vilket gjorde valet av databashanterare enkelt. Vidare hade också Cubelux utvecklare tidigare använt sig av MySQL och PHP.

PHP har använts som back-end av många applikationer med dynamiskt innehåll. Applikationen måste vara dynamisk för att visa olika data åt många användare på samma gång. Man hittar många rekommenderade ramverk efter en sökning på Google med sökorden “PHP MySQL framework”.

Av de ramverk många hade rekommenderat valde jag att testa två stycken. CodeIgniter och Symfony hade fått många bra recensioner och hade potentialen att driva applikationen. Ett test utfördes där jag utförde de förberedande uppgifterna från bådas hemsidor. CodeIgniter var den jag ansåg hade bäst potential efter att ha utfört uppgifterna. Symfony hade för många bibliotek jag inte behövde och hade för strikt kontroll över MVC-modellen. CodeIgniter hade mycket mindre funktioner men jag ansåg att man klarade sig med ett mera lättviktigt PHP ramverk.

Man behöver också ett lättförstått grafiskt gränssnitt för webbapplikationen. Jag kommer att använda sig av JavaScript biblioteken jQuery och Modernizr för att förkorta det grafiska finessernas utvecklingstid. Det är också bra praxis att använda dessa bibliotek med tanke på att man vill ha applikationen tillgänglig för flera olika plattformar och webbläsare. jQuery har också ett eget grafik bibliotek med färdiga koder för ofta förekommande element på en webbsida. Detta heter jQuery UI som också har använts i applikationen för att underlätta arbetet med ofta använda element.

## 2 METOD

När man utvecklar en dynamisk webbapplikation behöver man en databashanterare som kan hantera all data. MySQL är en fritt tillgänglig databashanterare och är den mest använda databas med öppen källkod i världen (Oracle Corporation 2013). MySQL finns tillgängligt för många plattformar och väljs ofta som databashanterare inom webbutveckling.

Kodarbetet är en mycket lång process ifall man börjar från början. Man kan undvika att “uppfinna hjulet på nytt” genom att använda sig av ett webbapplikationsramverk. Ramverket gör det enklare att ta fram och utveckla en webbapplikation. Utvecklingstiden blir också kortare för utvecklingen av det grafiska gränssnittet. Man har också lättare att återanvända funktioner man gjort i webbapplikationen eftersom man har möjligheten att referera till sina egna filer och funktioner i koden. Kraven för ramverket är att det skall vara baserat på PHP, det skall vara lättviktigt för att inte belasta servern i onödan och man skall lätt kunna lära sig ramverket.

Det ramverk man har valt är CodeIgniter som använder sig av Model-View-Controller mönstret för att hantera kontakten mellan databasen och applikationen eller applikationen och användaren (Ellislab, Inc 2012).

Grafiska finesser finns tillgängliga för alla webbläsare och JavaScript biblioteket Modernizr har underlättat kodarbetet genom att göra HTML5 och CSS3 tillgängligt för äldre webbläsare. Man kommer att använda jQuery eftersom det kan göra det grafiska gränssnittet smidigare. jQuery är ett populärt tillägg i webbutveckling för att förkorta kodarbetet med JavaScript. Det gör också informationshanteringen lättare genom att man med AJAX kan skicka data till PHP som i sin tur hanterar informationen enligt koden.

### 2.1 Databashanteraren

Databashanteraren MySQL:s popularitet på Internet gör det nästan till en standard inom IT och webbutveckling. Stora kända internet sajter med många miljoner användare och

komplexa databasstrukturer, som Facebook, Youtube och Wikipedia, har också gjort att databashanterarens användning i nya webbapplikationer har ökat (Oracle Corporation 2013).

Det flesta webbhotell har också installerat MySQL på sina servrar vilket gör det till ett naturligt val ifall man vill upprätthålla en webbapplikation med hög kapacitet billigt och tillgänglig för många användare. Man har också lagt till MySQL i mjukvarupaketprogrammen LAMP (Linux), MAMP (MAC OS X) och WAMP (Windows).

MySQL har också utvecklat ett eget mjukvaruprogram för att förenkla arbetet med databasen Detta har man kallat för MySQL Workbench. Programmet har ett grafiskt system man kan använda för att hantera information i databasen och visuellt utveckla en databasstruktur för MySQL. Man kan genom att exportera EER modellen som SQL satser få en databas med färdiga kopplingar. EER är forkorting av enhanced entity-relationship och är en utvidgad version av ER (entity relationship) modellen. Modellerna används för att visa hur tabellerna i databasen hör ihop. Man kan också använda MySQL Workbench för att säkerhetskopiera eller återställa sin databas (Oracle Corporation 2013).

## **2.2 Webbapplikationsramverk**

Webbapplikationsramverk används ofta idag för att förkorta utvecklingstiden för webbapplikationer. Företag har blivit mer krävande och vill ha sina applikationer snabbt och de behöver ofta väldigt dynamiska webbplatser.

En dynamisk webbplats kräver många databashämtningar och -inmatningar beroende på vem som använder webbplatsen för att visa och mata in information. Varje databasanrop utgör en säkerhetsrisk vilket man måste försöka förhindra med restriktioner för olika typer av användare, komplexa lösenord och filtrering av datainmatningar.

Med ett webbapplikationsramverk menas en samling bibliotek och klasser som man kan återanvända med hjälp av ramverket. Ofta förekommande bibliotek i PHP ramverk är databashanterings-, template- och sessionshanteringsbibliotek.

Nackdelarna med att använda ett ramverk är att ifall det finns ett fel i ramverket kan det påverka hela applikationen. Man ska också planera koden noga för att undvika redundans i ramverket. Det är lätt hänt att man skriver om samma funktion många gånger.

## 2.3 Model-View-Controller

Ramverk följer olika designmönster som programmeraren behöver förstå. Det är populärt att använda någon slags designmönster inom utveckling av webbapplikationer och andra mjukvaruimplementationer. Ett ofta använt mönster är Model-View-Controller (MVC) (Abeyasinghe 2009 s.30).

Ett MVC baserat ramverk kan vara väldigt strikt eller flexibelt i var man använder klasser. Vill man mata in eller hämta information skapar man en Modell (Model) var man skapar de funktioner man behöver för databashantering. En bra minnes regel är att man inte skall använda HTML taggar eller \$\_GET i en Modell fil. Modellen saknar presentations funktion i MVC mönster och den är inte alls kopplad direkt till HTTP förfrågningar.

Utvecklaren skapar en Vy (View) för att användaren fysiskt ska kunna se det innehåll som är menat för henne. Det är modellen som tar emot den data som ska presenteras för användaren. Vyn presenterar data som HTML, XML eller som ett epost meddelande för användaren. Man refererar också sina CSS- och JavaScript-filer i Vyn ifall man använder sig av sådana.

När användaren klickar på ett objekt på en sida byggt på ett MVC ramverk kör Controllern en funktion som säger till modellen vilken data som skall matas in eller visas för användare. Controllern kan också reagera på andra händelser än de användaren gör (Liu 2000 s. 119 - 121).

Genom att använda ett MVC baserat ramverk håller man sin kod snygg och prydlig och gör det lättare för andra utvecklare att förstå vad man gjort för vidare utveckling av ap-

plikationen. MVC ramverkens utvecklare har också det lättare att återanvända den egna koden.

## **2.4 CodeIgniter**

Det ramverk jag valt för att utveckla webbapplikationen för personalschemaläggning är CodeIgniter. Den aktuella versionen i skrivandets stund är 2.1.3. CodeIgniter är ett PHP ramverk med de nödvändiga biblioteken för webbapplikationen. Ramverket har en låg inlärningskurva och det är lätt att lösa problem som uppstår på vägen eftersom CodeIgniters dokumentation är mycket bra gjord.

### **2.4.1 Kort översikt av ramverket**

CodeIgniter är mycket lätt att komma igång med. Man behöver bara ladda ner filerna från CodeIgniters hemsida och packa upp paketet till webb rot katalogen. Applikationen är nu installerad på webbroten och man skall nu konfigurera config.php som finns i application/config/ för att säga åt ramverket vad webbservern har för adress. Det första steget av installationen är gjorda och nu kan man gå in och se applikationens välkomst vy med webbläsaren. (Ellislab, Inc 2012).

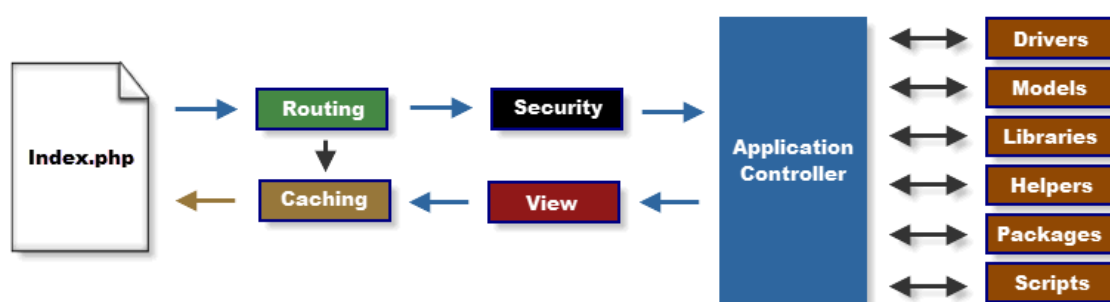
Första gången man ser filstrukturen i CodeIgniter är det svårt att förstå vilka filer man skall börja jobba med. De filer man mest kommer att arbeta med finns i "application"-katalogen. Konfigurationsfiler för applikationen återfinns i katalogen "config". Man hittar också en autoload.php i "config" katalogen. Autoload.php används för att ladda ofta använda bibliotek globalt för applikationen. En annan viktig fil är database.php som man måste konfigurera ifall man vill använda en databas med ramverket.

Det finns tre kataloger där man hittar sina MVC relaterade filer. Katalogerna heter inte alltför överraskande "models", "views" och "controllers". Dessa är skapade för att man lättare ska kunna hålla koll på sina filer i sin applikation. Genom att öppna mapparna och granska filerna får man en första insikt i hur ramverket är uppbyggt. Filerna man skapar i dessa mappar måste vara PHP filer för att man skall kunna använda sig av dem.



### 2.4.2 MVC för att organisera applikationen

CodeIgniter använder sig av MVC för att det skall gå snabbare och lättare att utveckla applikationer. Ramverkets tre kataloger har man gjort för att man lätt skall kunna navigera i sina applikationsfiler. Det är bra att man redan från början försöker vara strikt med att endast placera kod för presentation i "views" -katalogen, funktioner för att kontrollera applikationens dataflöde i "controllers" -katalogen och applikationens logik och databas förfrågningarna i "models" -katalogen. På det här sättet slipper man skapa mycket ny kod ifall man behöver ett par nya vyer.



Figur 1. CodeIgniters flödesschema (EllisLab, Inc 2012)

### 2.4.3 Autentiseringsbiblioteket Ion Auth

När man loggar in skall applikationen kontrollera ifall man är en Administrator eller en anställd. För att åstadkomma en rollautentisering använder vi oss av ett färdigt byggt bibliotek för inloggning vid namn Ion Auth, vilket är under ständig utveckling. Ion Auth innehåller många olika funktioner såsom inloggning, utloggning, användarregistrering, användaruppdatering, användarbortagning, glömt lösenord, sessionshantering för inloggade användare och rollhantering.

För att använda sig av biblioteket måste man först kopiera filerna, som man har fått med sig när man laddade ner paketet, till sin applikation och skapa databastabeller som Ion Auth kräver. En SQL-fil finns med i paketet som skapar de nödvändiga tabellerna.

### 2.4.4 CodeIgniters mål

Man har skapat ramverket enligt CodeIgniter för att "maximera prestandan, kapaciteten och flexibiliteten i det minsta och lättaste paketet". CodeIgniters komponenter laddas

när man anropar dom vilket gör att man inte initierar komponenterna globalt. Systemet förutspår ingenting om vad som behövs för applikationen förutom de centrala funktionerna i ramverket, vilket gör systemet väldigt resurssnålt från början. Man har gjort ramverket med “Loose Coupling”, vilket innebär att ju mindre komponenterna i ett ramverk beror på varandra destomera återanvändbart och flexibelt blir det. Singularitet i ramverkets klasser och funktioner gör att man kan utnyttja det maximalt (Ellislab, Inc 2012).

## **2.5 JavaScript**

JavaScript är ett skriptspråk utvecklat av Netscape som används av miljontals webbsidor över hela världen. Det är ett dynamiskt skriptspråk som stöder prototyp-baserad objekt konstruktion. Syntaxen liknar både Java och C++ för att göra det lättare att börja använda språket. Man kan använda JavaScript som ett procedurellt och objektorienterat språk.

HTML Document Object Model (DOM) element kan styras med JavaScript, vilket betyder att man kör språkets funktioner med webbläsarens JavaScript motor. Därför används språket mycket på klientsidan för att man t.ex. skall få nytt innehåll på sidan utan att behöva ladda om sidan (Benedetti & Cranley 2011 s. 7).

När man vill använda JavaScript på en webbsida inkluderar eller bäddar man in JavaScript koden i HTML-sidor. Java och JavaScript har inget med varandra att göra. Den enda likheten är att de använder samma syntax som C (Mozilla Developer Network 2013).

### **2.5.1 jQuery**

W3Techs Web Technology survey påstår att över 55 % av de en miljon populäraste webbsidorna använder sig av jQuery och var av 39 % inte använder sig alls av JavaScript (Q-Success 2013).

Biblioteket skapades av John Resig och första versionen släpptes på BarCamp NYC 2006. Den aktuella versionen i skrivandets stund är 1.9.1. jQuery är ett objektorienterat JavaScript ramverk, som levereras med HTML sidor och körs i webbläsaren. Man an-

vänder jQuery för att lösa många cross-browser problem som kan uppstå i och med att man vill ge användaren en professionell upplevelse av webbsidan.

Det finns nu stöd för webbläsarna Internet Explorer 6+, Chrome, Firefox, Safari och Opera. Biblioteket är utrustat med eleganta och kraftfulla JavaScript lösningar för att göra webbsidans beteende mera interaktivt. Det har lett till att man har mycket snyggare webbsidor eftersom det inte krävs att front-end utvecklaren är expert på JavaScript. Användare slipper också använda sig av tredjeparts program som Adobe Flash eller Microsoft Silverlight för att göra avancerade grafiska animationer.

Det man behöver hjälp med att animera på sina webbsidor åstadkommer man väldigt lätt med jQuery. jQuery har många olika animationer i sitt animationsbibliotek man kan använda. Man har också tillgång till “drag and drop” funktioner i jQuery.

Från och med version 1.3 har jQuery använt sig av en CSS selektor motor vid namn Sizzle (Resig 2009). JavaScript motorn hjälper jQuery användarna genom att lägga till stöd för CSS selektorer i jQuery. Man har tillgång till stöd för CSS3 selektorer (Sizzle 2013). Genom att använda selektorer blir det lättare för utvecklare att välja rätt DOM element med mindre kod.

### **2.5.2 Modernizr**

Modernizr är ett litet JavaScript bibliotek som används för att hitta vilka tillgängliga CSS3 och HTML5 funktioner som är tillgängliga i användarens webbläsare. Detta kallas för “feature detection” vilket är snabbare än att kontrollera vilken browser som används. Helt enkelt kollar Modernizr ifall din browser har stöd för den sökta funktionen ursprungligen. Man försöker få ett slut på det så kallade “UA sniffing” för att göra ett mera tillförlitligt mekaniskt tillvägagångssätt för att veta vad man kan göra och inte göra med webbläsaren.

Modernizr testar över 40 “next-generation” funktioner på några millisekunder. Det skapar ett JavaScript objekt kallat Modernizr som innehåller ett resultat av de utförda testerna som boolean attribut. Genom att lägga till CSS-klasser i HTML elementet så får

man veta vad som stöds och inte stöds av webbläsaren. Ett script kan laddas för att lägga till “polyfills” för att ge stöd för äldre webbläsare (Modernizr 2013).

## 3 IMPLEMENTATION

### 3.1 Databasstrukturen

Utvecklingen av webbapplikationen började planerades på ett vanligt papper. Det man använde som modell för schemat var en vanlig fickkalender. Fickkalender visade en vecka per uppslag med klockslagen 6 - 21.

Den första självklara tabellen var den innehållande informationen om personalen. Man sökte den kritiska informationen som skall sparas för personal. Namn, adress, användarnamn, e-postadress och telefonnummer var den information man ansåg behövdes. Det ritades upp kolumner med samma namn som informationen man behövde på pappret. Tabellen behövde också fälten från Ion Auth så de sattes också till för att biblioteket skulle fungera som det ska.



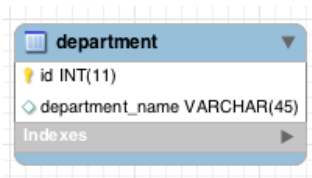
The image shows a screenshot of a database schema viewer for a table named 'users'. The table has the following columns and data types:

Column Name	Data Type
id	MEDIUMINT(8)
ip_address	VARBINARY(16)
username	VARCHAR(100)
password	VARCHAR(80)
salt	VARCHAR(40)
email	VARCHAR(100)
activation_code	VARCHAR(40)
forgotten_password_code	VARCHAR(40)
forgotten_password_time	INT(11)
remember_code	VARCHAR(40)
created_on	INT(11)
last_login	INT(11)
active	TINYINT(1)
first_name	VARCHAR(20)
last_name	VARCHAR(20)
phone	VARCHAR(15)
adress	VARCHAR(45)
postalcode	VARCHAR(45)
city	VARCHAR(45)

Below the columns, there is a section for 'Indexes' which is currently empty.

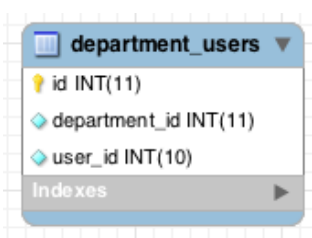
Figur 2. Databastabellen users

Avdelningstabellen kommer att innehålla information om avdelningen. **Department** innehåller bara två kolumner. Den första kolumnen id är till för att koppla ihop användare med rätt avdelning och den andra är för namnet på avdelningen.



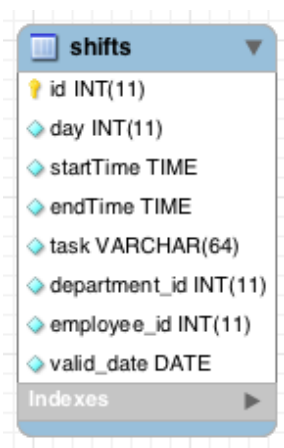
Figur 3. Databastabellen department

När jag gjort klart tabellerna **users** och **department** märkte jag att man måste koppla ihop de på något sätt. En **user** kan ha flera **department** samt en **department** kan ha flera **users**. Jag bestämde mig för att skapa en tabell som heter **department\_users** som håller koll på vilka avdelningar som innehåller vilka användare.



Figur 4. Databastabellen department\_users

En tabell behövs också för att visa vilka arbetsskiften som skall visas för olika användare beroende på vilka avdelningar en anställd kan jobba. Varje person kommer att kunna vara kopplad till tabellen **department** varifrån användaren får information om vilka skiften som finns att välja på i avdelningen. Genom att göra en egen tabell för arbetsskiften gör man det lättare att koppla ihop flera skiften med en avdelning.



Figur 5. Databastabellen shifts

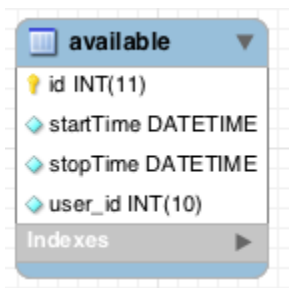
Alla skiften som är bokade för alla användare vill man kunna spara i en tabell för att visa varje månads planerade arbetstider för administratörerna. Ett knapptryck av admini-

stratorn uppdaterar en tabell med kolumner för informationen man behöver för att kunna visa månadens anmälda arbetstider. Programmet skall vidareutvecklas för att automatiskt uppdatera tabellen ifall en anställd gör en ändring i sina arbetstider.



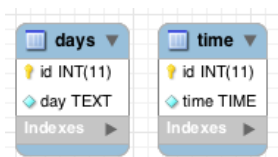
Figur 6. Databastabellen plannedtime

Man behöver också en tabell för att spara information om en deltidsanställd vill anmäla vilka dagar den kan arbeta och på detta sätt visa alla skiften tillgängliga för den anställda. Tabellen kommer också att presenteras i en skild vy för administratör för att lätt se tillgängligheten av deltidsanställda. Tabellen används också för att filtrera arbetstider ifall en heltidsanställd har anmält sig frånvarande någon dag den månaden.



Figur 7. Databastabellen available

Tabeller för de olika klockslagen och dagarnas namn gjordes också för att man lättare skulle se vilken dag man har bokat olika skiften. Klockslagstabellen gjordes för att man kan ha mindre data i PHP filerna.



Figur 8. Databastabellen days and time

## 3.2 Gränssnittet

Det viktigaste med applikationen är att alla vyer skall visa den viktigaste informationen direkt åt användaren snyggt och simpelt. Applikationen kommer också att göras responsiv senare. Responsiv betyder att man gör grafiska förändringar på basis av populära skärmapplösningar. Jag har använt W3Schools.com statistik över populära skärmapplösningar (W3Schools.com 2013) för att bestämma sig för att bredden på applikationen kommer, under utvecklingstiden, vara 960 pixlar.

Gränssnittet kommer att användas av två användargrupper. Användargrupperna kommer att behöva skilda vyer eftersom information som anses vara av hög prioritet är olika för dessa. Man behöver också en portal för att styra användarna till rätt vy. Portalen kommer att vara en inloggningssida som styr användarna till en sida för administrering av arbetstider. Användarna kommer att vara Administrator och Employee. De kommer att få varsin administrerings vy med element som visar innehåll från databasen.

Användare med Employee rättigheter kommer att kunna boka tillgängliga arbetstider åt sig själv. Man behöver ha möjlighet att anmäla sig tillgänglig eller frånvarande i systemet och man behöver också en möjlighet att kunna uppdatera sin tillgänglighet eller valda sina arbetstider. Den fjärde saken Employee användare behöver är att uppdatera sin användarinformation.

Administrator användare kommer att ha möjlighet att hantera alla användare. Dessa användare kommer också att göra skiftesbokningar för sig själva eftersom man har antagit att administratorer är anställda på företaget. Administrator vyn kommer att ha behov av att visa flera olika vyer på basis av databastabellerna och kommer där med att kräva mera utvecklingstid.



## 3.3 Informationshanteringen

### 3.3.1 Hanteringen av avdelningar

Det första man måste göra är att lägga till avdelningar. En avdelning kräver inte mera än ett namn i alfa versionen. Avdelningar läggs till i applikationen genom att fylla i ett fält för avdelningens namn.

Jag började med att skapa en vy med de fält som krävdes för att kunna skicka data till databasen. CodeIgniter kommer med ett bibliotek för olika formulär varifrån man kan kalla på de mest använda formulärelementen. Ett formulär skapas genom att “öppna” formuläret och lägga till de element man vill använda och sedan “stänga” formuläret. Öppningsfunktionen i formuläret är `form_open('controller/function')`. Det man måste fylla i är den controller och den funktion i kontrollern man vill använda sig av. CodeIgniter sköter om att skapa en länk som kör funktionen man valt. Namnet på avdelningen ifylles i ett fält som skapas med `form_input('namn_på_fältet')`. Formulär funktionen skapar ett input fält med ett namn för inmatningsfältet. Det sista elementet vi behöver i formuläret är en “submit” -knapp som skickar informationen i inmatningsfältet till den funktion man valt i `form_open()`. Den sista funktionen man behöver i formuläret är `form_close()` vilken skapar avslutningstaggen för formuläret.

När man klickar på knappen skickas informationen till databastabellen och funktionen i kontrollern avgör vad som skall hända med informationen. Informationen kontrolleras med hjälp av CodeIgniters Form Validation klass som har en funktion `set_rules()` man kan använda för validering av data. När informationen är godkänd av funktionen `set_rules()` använder man funktionen `run()` i en IF- sats för att koden skall veta ifall informationen skall sparas i databasen eller ifall den skall skicka ett felmeddelande till användaren. Ifall informationen är godkänd och sparad i databastabellen syns avdelningen i en vy man skapat för att visa de avdelningar som finns i databastabellen.

Vyn för avdelningarna visar en tabell med avdelningens namn, alla anställda som hör till avdelningen och en länk där man kan ta bort avdelningen. Länken för att ta bort en

avdelning är kopplad till en funktion i Admin -controllern som tar bort avdelningen på basis av id i avdelningstabellen.

Avdelningsnamnet är en länk till en vy för att kunna ändra på informationen för avdelningen. Om man vill ändra namnet på en avdelning klickar på avdelningsnamnet i den tabellen i vyn där alla avdelningar är listade. Man blir slussad till en editerings vy för avdelningar. Vyn är gjord på likadant sätt som för att lägga till avdelningar med den skillnaden att den valda avdelningens namn har hämtats från databasen. Formuläret använder sig av en annan funktion som uppdaterar en rad i avdelningstabellen på basis av avdelningens id, som hämtas från adressfältet, istället för att skapa en ny rad.

Den sista länken kallar på en funktion i Admin -kontrollern. Funktionen tar på basis av det tredje segmentet i adressfältet, vilket är avdelningens id, bort en rad i databasen. CodeIgniters URI klass har en funktion `segment (n)` vilken hämtar det n-te segmentet ur adressfältet.

Avdelningsvyns tabell får sin information från en variabel som skickats från kontrollern. Variabeln innehåller en array med information om avdelningen som hämtats från databasen via en modell. Modellen innehåller en funktion som hämtar alla rader i avdelningstabellen och kopplar ihop alla användare som jobbar på avdelningen. Funktionen innehåller en MySQL query som kopplar ihop tabellerna **department**, **department\_users** och **users** för att få den data man vill se i tabellen. Jag har använt mig av CodeIgniters databas bibliotek när jag gjort MySQL queryn. Funktionen jag använt mig av är `$this->db->query()` för att skapa queryn. Koden loopar också genom varje rad och gör dom till variabler i en array för att kunna presentera informationen i databasen för användaren.

### 3.3.2 Hantering av användare

Administratören är den som har rätt att hantera alla användare i applikationen. Användarna är de som är anställda på företaget och behöver möjlighet till att se sina arbetstider. Vyn som visar informationen om anställda åt administratören hämtar sin information över anställda från databastabellen `users`. Jag har använt Ion Auth biblioteket vid hanteringen av användarinformation för att göra applikationen säkrare. Ion Auth kom med ett skript för att skapa tabellerna i databasen som behövs för att Ion Auth ska fungera.

Tabellerna som skapades för användare hade inte från början alla kolumner man behövde i applikationen, men jag satte till de kolumner man behövde efteråt.

Vyn används för att skapa, editera eller ta bort användare. De kolumner som kommer att visas är namn, telefonnummer, e-post, adress, avdelning och ta bort anställd. Jag har på liknande sätt som i avdelningsvyns tabell hämtat informationen via Admin -controllern och en funktion som hämtar informationen från flera databas tabeller. Jag har kopplat ihop tabellerna med en liknande databas funktion som jag gjort för avdelningarna.

Anställda kan läggas till genom att klicka på en länk i vyn för anställda. Administratören skickas till en vy för att lägga till anställda när man klickat på länken. Fälten man måste fylla i är förnamn, efternamn, e-post, telefonnummer, gata, postnummer, stad, lösenord och en upprepning av lösenordet. Administratören väljer ett lösenord åt användaren som man kan byta själv när han loggar in första gången.

Vyn är skapad med samma CodeIgniter formulär funktioner som för att lägga till avdelningar. Kontrollen för användarinformation är väldigt viktig för att man skall hindra intrång i applikationen. Ion Auth paketet innehöll färdiga funktioner för att lägga till användare i users tabellen. Jag gjorde små ändringar i funktionen för att också de tillagda kolumnerna skulle bli kontrollerade på rätt sätt. Användaren sparas i tabellen users, ifall användardatan blir godkänd av Ion Auth bibliotekets kontroll, med en funktion från Ion Auth biblioteket för att salta lösenordet på rätt sätt.

När användaren sparas skickas man till en vy för att lägga till användare i en avdelning. Denna vy kommer man åt endast genom att lägga till en användare i databasen. Vyn visar alla avdelningar och har två kolumner. Den ena kolumnen visar avdelningens namn och den andra innehåller en länk för att lägga till användaren till en avdelning. När man valt avdelning skickas man tillbaka till vyn som listar alla anställda.

Genom att klicka på en anställds namn kan man editera informationen för den anställda. Editeringsvyn ser likadan ut som när man lägger till en anställd. När man editerar en anställd och sparar sin information uppdateras informationen med hjälp av Ion Auth bibliotekets `update()` funktion. Man kan lägga till eller ta bort en anställd från olika avdelningar genom att klicka på någon av avdelningarna i avdelningskolumnen. Vyn för att lägga till en anställd till avdelningen visar alla avdelningar men man kan endast

lägga till den anställda i de avdelningar som han inte är anmäld till. Här kan man också ta bort en anställd från avdelningen, men endast från de avdelningar där den anställda finns listad. Användaren kommer att kunna boka arbetsturer som finns listade för de avdelningar man är insatt i.

### 3.3.3 Hanteringen av skiften

Skiften innehåller information om vilken avdelning, vilken dag, starttid, sluttid, ifall det är ett tillfälligt skifte eller inte och ifall skiftet hör till en anställd. Informationen visas för administratören i vyn för skiften. Man börjar med att lägga till ett skifte genom att klicka sig vidare till vyn för att lägga till ett skifte.

Först skall man välja ifall det är ett permanent skifte eller inte. Utgångsläget är att skiftet är permanent. Ett permanent skifte måste ha en anställd och den anställda kan väljas från en dropdown -meny. Man har skapat dropdown -menyn med CodeIgniters formulärfunktion `form_dropdown()`. Alla anställda är listade i menyn och dessa hämtas från databasen via en funktion i Admin -controllern. Funktionen tar kontakt med modellen som hanterar informationsflödet i **users** tabellen. Här har jag skapat en funktion som hämtar utvalda kolumner och alla rader i databastabellen **users**. Variabeln för användar arrayen kunde inte användas direkt i dropdown funktionen utan behövdes arbetas om i vyn för att lägga till skiften eftersom man vill se namnen på de anställda och inte deras id nummer. Jag har använt samma teknik när jag har gjort resten av dropdown-menyerna i vyn. Efter att man valt anställd måste man välja den dag skiftet skall gälla. Dagarna hämtas från **days** tabellen.

Ett skifte som inte är permanent visar istället för anställda ett fält för datum. Man kan inte själv skriva in ett datum för att eliminera risken att datumet blir fel skrivet. När man klickar i fältet visas jQuery UI:s datumväljare för administratören. Här kan administratören välja ett datum ur en kalender. Administratören kan välja ett datum som är nyare än dagens datum och tre veckor framåt. Man har också tagit bort möjligheten att välja en helgdag ur kalendern och inaktiverat dropdown menyn för att välja vilken dag skiftet skall gälla. En JavaScript funktion ser till att det visas rätt dag i fältet med namn på dagen.

Nästa steg är att fylla i vad för uppgift skiftet står för. Jag har valt att göra en textruta av uppgiften ifall uppgiften ändras. Skiftet behöver också en avdelning som det skall vara kopplat till. Här har man listat alla avdelningar som finns i **departments** tabellen i dropdown -menyn. Skiftets starttid och sluttid får klockslagen från **times** tabellen. Jag har i alpha versionen bara listat hela klockslag men det finns möjlighet att lägga till vilka klockslag man vill i tabellen. Administratören kan nu spara skiftet genom att klicka på Save och han skickas till vyn för skiften och ser att skiftet han skapat finns listat där. Man har möjlighet att editera ett skifte i vyn för skiften genom att klicka på länken "Edit shift". Vyn för att editera skiften är likadant gjord som den för att lägga till skiften. Skillnaden är att skiftets data är färdigt ifyllt i editeringsvyn för att göra editeringen lättare för administratören. Administratören har behov av att radera skiften och det kan han göra genom att klicka på länken "Remove shift".

### **3.3.4 Hanteringen av planerad arbetstid**

Vyn för planerad arbetstid visar en tabell med alla anställdas bokade skiften. Först visas alla permanenta skiften och sist visas icke permanenta skiften. Administratören har möjlighet att editera och ta bort icke permanenta skiften i denna vy. Man har tänkt sig att det är bara anställda som skall boka skiften för att lätta på administratörns jobb. Listan visar veckodag, starttid, sluttid, avdelning, uppgift, ägare av skiftet och vilket datum skiftet är ifall det inte är ett permanent skifte.

### **3.3.5 Hanteringen av skiften för anställda**

Den anställdas vy för att hantera arbetstid visar endast den arbetstid som är bokad för denne. När en anställd med permanenta skiften öppnar applikationen kommer denne att se alla skiften som är bokade åt honom. En anställd som inte har permanenta skiften kommer först att mötas av en tom tabell. En anställd som inte har permanenta skiften måste först klicka sig till bokningsvyn för att kunna se sina arbetstider. Vyn visar veckodag, starttid, sluttid, datum

## 4 RESULTAT

### 4.1 Template

Varje Controller som presenterar en vy kan innehålla ett objekt för att bestämma vad som t.ex. ska stå inne i titel taggen. Genom att skapa en variabel och tillhandahålla nycklar med bestämda värden i controllern kan man styra vad som ska stå i titel taggen. Titelns värde visas i vyn när man skriver nyckelns namn i den mall man vill använda (Ellislab, Inc 2013).

Detta har också kommit till användning när man vill göra vyerna. Jag har genom att använda nycklar i variabeln för vilken vy som skall användas när en controller laddas minskat på koden i applikationen. Applikationen använder nyckeln *main\_content* för att styra vilken vy som skall laddas från controllern.

När man vill använda nycklarna i sin applikation för att visa det man behöver åt användaren använder man sig av CodeIgniters vy laddnings funktion.

```
function login()
{
    $this->data['title'] = "Login";
    $this->data['main_content'] = "auth/login";
    $this->load->view('base/template', $this->data);
}
```

Figur 9. Kodexempel på hur man kan visa en vy i CodeIgniter

```
<head>
    <title><?php echo $title;?></title>
</head>
```

Figur 10. Kodexempel på hur man kan visa en text sträng ur en nyckel

Jag har från början skapat vyer för vanliga elementet i webbapplikationen. Dessa bas vyer finns i en PHP fil man har kallat för template.php. Jag har genom att återanvända dessa view element undvikt att skriva samma kod på olika ställen. Funktionen för att göra vyer åt användaren kan använda sig av en variabel för att göra sidan dynamisk.

```

<?php $this->load->view('base/header'); ?>

<section id="main">
    <?php $this->load->view($main_content); ?>
</section>

<?php $this->load->view('base/footer'); ?>

```

Figur 11. Kodexempel på hur man skapat en bas mall

## 4.2 Inloggningsportalen

Applikationens inloggningsportal kräver endast att man skall kunna skriva in sitt användarnamn, lösenord och klicka på “Logga in” för att man ska skickas vidare till administreringsvyn. Före man skickas vidare till någon av vyerna behandlar autentiseringsbiblioteket med rollhantering användarinformationen. Biblioteket man har använt heter Ion Auth. Ion Auth är mycket lättanvänt och jag har modifierat det så man kan använda det med den egna applikationen.

```

//redirect if needed

function index()
{
    if (!$this->ion_auth->logged_in())
    {
        //redirect them to the login page
        redirect('auth/login', 'refresh');
    }
    elseif (!$this->ion_auth->is_admin())
    {
        //sends worker to worker dashboard if logged in
        redirect('employee/dashboard', 'refresh');
    }
    elseif ($this->ion_auth->is_admin())
    {
        //sends admin to admin dashboard if logged in
        redirect('admin/dashboard', 'refresh');
    }
    else
    {
        //set the flash data error message if there is one
        $this->data['message'] = (validation_errors()) ? validation_errors() : $this->session->flashdata('message');

        //list the users
        $this->data['users'] = $this->ion_auth->users()->result();
        foreach ($this->data['users'] as $k => $user)
        {
            $this->data['users'][$k]->groups = $this->ion_auth->get_users_groups($user->id)->result();
        }

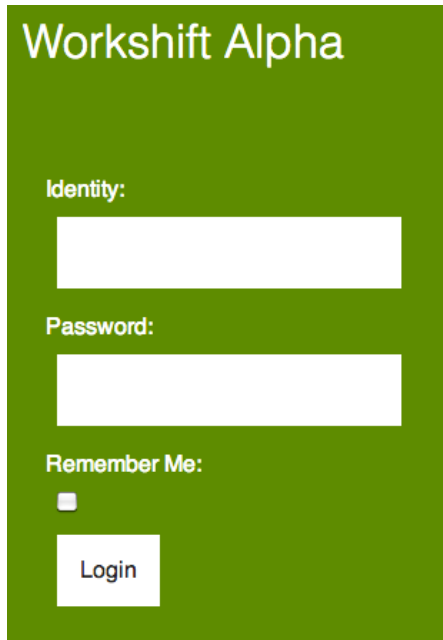
        $this->load->view('auth/index', $this->data);
    }
}

```

Figur 12. Ett exempel ur autentiserings kontrollern

Det man gör efter autentiseringen är att skicka användaren till den rätta vyn så användaren ska snabbt få tillgång till den information ämnat för henne. Inloggningsvyn består av två input fält, en knapp för att logga in och en checkbox för att spara användarens

inloggningsuppgifter på den använda datorn. När man trycker på knappen skickas den registrerade användaren till rätt vy och den oregistrerade hänvisas till samma webbplats. Administratören måste ha skapat användaren i systemet innan man kan logga in. Har man tappat bort sitt lösenord måste man ta kontakt en administrator för att få ett nytt.



Workshift Alpha

Identity:

Password:

Remember Me:

Login

Figur 13. Inloggningsportalens vy

Det som händer när man trycker på knappen är att kontrollern kontrollerar ifall båda inputfälten har ett värde. Har båda fälten ett värde som överensstämmer med en rad i användartabellen skickas användaren tillbaka till index funktionen i Autentiseringskontrollern för att slussas till den rätta vyn. Vyn laddas från den angivna stigen med CodeIgniters funktion `$this->load->view()`. Om man glömt att fylla i något av fälten uppmanas man fylla i den saknade informationen. Den användare som kommit ihåg att fylla i båda fälten men med värden som inte överensstämmer med en rad i tabellen blir uppmanad att fylla i rätt värden.



```

function login()
{
    if ($this->ion_auth->logged_in() && !$this->ion_auth->is_admin()){
        //sends worker to worker dashboard if logged in
        redirect('employee/dashboard', 'refresh');
    }
    elseif($this->ion_auth->is_admin() && $this->ion_auth->logged_in()){
        //sends admin to admin dashboard if logged in
        redirect('admin/dashboard', 'refresh');
    }
    else{
        $this->data['title'] = "Login";
        $this->data['main_content'] = 'auth/login';

        //validate form input
        $this->form_validation->set_rules('identity', 'Identity', 'required');
        $this->form_validation->set_rules('password', 'Password', 'required');

        if ($this->form_validation->run() == true){
            //check to see if the user is logging in
            //check for "remember me"
            $remember = (bool) $this->input->post('remember');

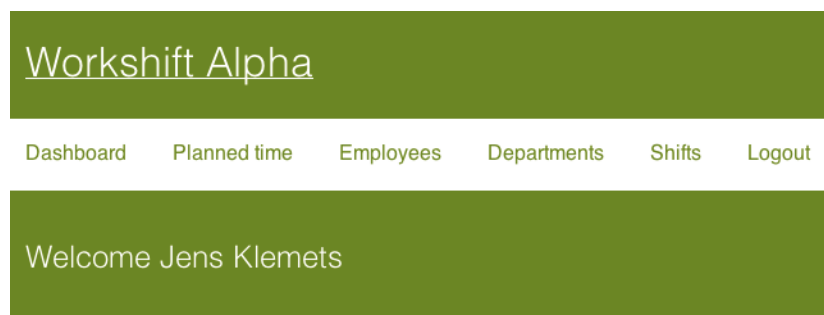
            if ($this->ion_auth->login($this->input->post('identity'), $this->input->post('password'), $remember)){
                //if the login is successful
                //redirect them back to the home page
                $this->session->set_flashdata('message', $this->ion_auth->messages());
                redirect('/', 'refresh');
            }
            else{
                //if the login was un-successful
                //redirect them back to the login page
                $this->session->set_flashdata('message', $this->ion_auth->errors());
                redirect('auth/login', 'refresh'); //use redirects instead of loading views for compatibility with MY_Controller libraries
            }
        }
        else{
            //the user is not logging in so display the login page
            //set the flash data error message if there is one
            $this->data['message'] = (validation_errors()) ? validation_errors() : $this->session->flashdata('message');
            $this->data['identity'] = array('name' => 'identity',
                'id' => 'identity',
                'type' => 'text',
                'value' => $this->form_validation->set_value('identity')
            );
            $this->data['password'] = array('name' => 'password',
                'id' => 'password',
                'type' => 'password'
            );
            $this->load->view('base/template', $this->data);
        }
    }
}

```

Figur 14. Kodexempel från inloggningsportalens kontroller

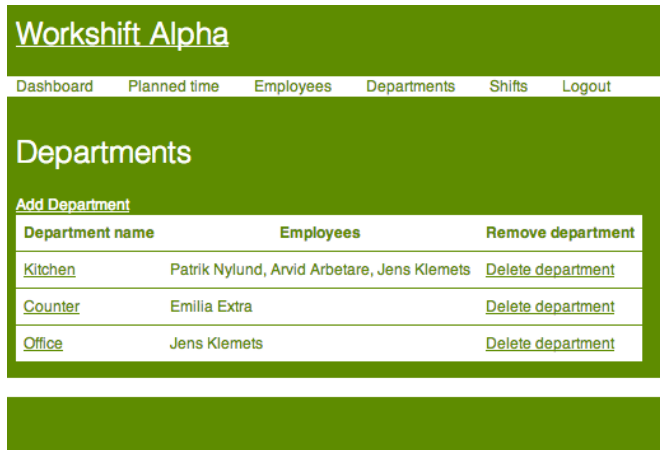
## 4.3 Administrator vyn

Jag har skapat fem huvudvyer för Administratören. Dessa vyer är till för att hantera avdelningar, anställda och arbetstider. Den första vyn har man valt att kalla “Dashboard” där administratören skall kunna se utdrag från de olika delarna i applikationen. Man har tänkt att kunden skall få en skräddarsydd Dashboard och har därför inte i alfa versionen skapat utdragen.



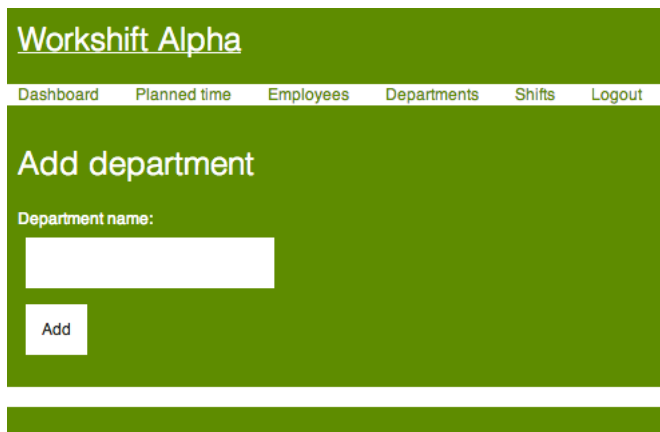
Figur 15. Administrator vyns Dashboard

Den andra vyn är för hanteringen av avdelningar. Här visar man en tabell åt Administratörn där han kan se alla avdelningar, avdelningarnas anställda och länkar för att redigera eller ta bort avdelningar. Ovanför tabellen finns en länk som tar administratörn till en vy för att lägga till avdelningar.



Department name	Employees	Remove department
<a href="#">Kitchen</a>	Patrik Nylund, Arvid Arbetare, Jens Klemets	<a href="#">Delete department</a>
<a href="#">Counter</a>	Emilia Extra	<a href="#">Delete department</a>
<a href="#">Office</a>	Jens Klemets	<a href="#">Delete department</a>

Figur 16. Avdelnings vyn



Workshift Alpha

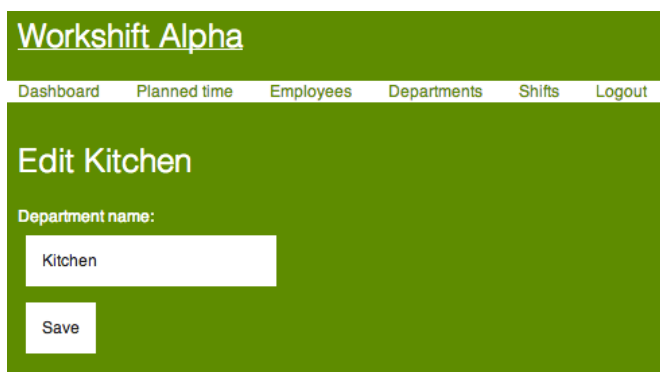
Dashboard Planned time Employees Departments Shifts Logout

### Add department

Department name:

Add

Figur 17. Vyn för att lägga till en avdelning



Workshift Alpha

Dashboard Planned time Employees Departments Shifts Logout

### Edit Kitchen

Department name:

Save

Figur 18. Vyn för att editera en avdelning

Vyn för hantering av användare visar en tabell med information om de anställda och länkar för att redigera och ta bort anställda. Här har jag också satt till en länk för att lägga till anställda. När man lägger till en anställ skickas administratören till en vy för att lägga till den anställda i en avdelning och sen tillbaka till vyn med informationen om de anställda.

Workshift Alpha						
Dashboard Planned time Employees Departments Shifts Logout						
Employees						
Add employee						
Name	Phone	Email	Adress	Department	Delete employee	
<a href="#">Jens Klemets</a>	23465	jens@cubelux.fi	Internetway 2 343434 Cyberspace	<a href="#">Office</a>	<a href="#">Delete employee</a>	
<a href="#">Patrik Nylund</a>	0501234567	patrik@workshift.com	Smedsbyvägen 4 344363 Smedsby	<a href="#">Kitchen</a>	<a href="#">Delete employee</a>	
<a href="#">Arvid Arbetare</a>	0501234567	arbetare@workshift.com	Arbetaregatan 5 555555 Arbetsstad	<a href="#">Kitchen</a>	<a href="#">Delete employee</a>	
<a href="#">Emilia Extra</a>	3425345324	extra@workshift.com	asdfgs 6 345342 sdfgsdfg	<a href="#">Counter</a>	<a href="#">Delete employee</a>	

Figur 19. Vyn som visar alla anställda

Workshift Alpha	
Dashboard Planned time Employees Departments Shifts Logout	
Add Employee	
First Name:	<input type="text"/>
Last Name:	<input type="text"/>
Email:	<input type="text"/>
Phone:	<input type="text"/>

Figur 20. Vyn för att lägga till en ny anställd

Workshift Alpha

Dashboard Planned time Employees Departments Shifts Logout

Choose department to add user in:

Departments	Add Employee	Remove Employee
Kitchen	<a href="#">Add</a>	<a href="#">Remove</a>
Counter	<a href="#">Add</a>	<a href="#">Remove</a>
Office	<a href="#">Add</a>	<a href="#">Remove</a>

Add new employee

Figur 21. Vy för att lägga till en anställd i en avdelning

Workshift Alpha

Dashboard Planned time Employees Departments Shifts Logout

Edit Information

Please enter the users information below.

First Name:

Jens

Last Name:

Klemets

Phone:

23465

E-mail:

jens@cubelux.fi

Figur 22. Vyn för att editera anställd

Arbetsshiftsvyn visar en tabell med information om de skiften som finns tillgängliga. Tabellen innehåller också länkar för att lägga till och ta bort skiften. Man lägger till skiften genom att klicka på en länk som skickar en vidare till en vy där man skall fylla i information om skiftet. Här kan man välja ifall skiftet är permanent eller inte. Ett permanent skifte betyder att det är dedikerat till en anställd och inte kan bokas av andra anställda. När man sparar skiftet skickas man till vyn med tabellen över skiften.

Workshift Alpha									
Dashboard Planned time Employees Departments Shifts Logout									
Shifts									
Add Shift									
Day	Start time	End time	Department	Task	Shift Owner	Valid date	Edit shift	Remove shift	
Monday	07:00	18:00	Kitchen	Barista	Arvid Arbetare	Permanent	<a href="#">Edit Shift</a>	<a href="#">Remove Shift</a>	
Monday	08:00	16:00	Kitchen	Dish washer	Patrik Nylund	Permanent	<a href="#">Edit Shift</a>	<a href="#">Remove Shift</a>	
Monday	12:00	15:00	Counter	Rush hour help		2013-03-04	<a href="#">Edit Shift</a>	<a href="#">Remove Shift</a>	
Monday	09:00	17:00	Office	Manager	Jens Klemets	Permanent	<a href="#">Edit Shift</a>	<a href="#">Remove Shift</a>	

Figur 23. Vyn som presenterar alla skiften i applikationen

Workshift Alpha	
Dashboard Planned time Employees Departments Shifts Logout	
Add shift	
Permanent shift: <input checked="" type="checkbox"/>	
Employee: <div> <div>✓ Jens Klemets</div> <div>Patrik Nylund</div> <div>Arvid Arbetare</div> <div>Emilia Extra</div> <div>Jens Klemets</div> </div>	
Task: <input type="text"/>	
Department: <input type="text" value="Kitchen"/>	
Start time: <input type="text" value="07:00:00"/>	
End time: <input type="text" value="18:00:00"/>	
<input type="button" value="Add"/>	

Figur 24. Vy för att ge en anställd ett skifte

Workshift Alpha

Dashboard Planned time Employees Departments Shifts Logout

### Add shift

Permanent shift:  
☐

Choose valid date:

Day:

Task:

Department:

Start time:

End time:

Figur 25. Vy för att skapa ett tillfälligt skifte

Workshift Alpha

Dashboard Planned time Employees Departments Shifts Logout

### Add shift

Permanent shift:  
☐

Choose valid date:

Day:

Task:

Department:

Start time:

End time:

Figur 26. Vy för att skapa ett tillfälligt skifte med kalender

Den sista vyn visar all planerad tid. Denna vy är endast till för att se alla bokade skiften. Administratörn kan boka ett skifte åt sig själv genom att klicka på en länk som för honom vidare till vyn för att boka skiften, men han kan inte boka åt någon annan användare. Man har dock rätt att ta bort ett bokat skifte eftersom den anställda inte kan avboka ett skifte från sin vy.

Workshift Alpha							
Dashboard Planned time Employees Departments Shifts Logout							
Planned time							
Weekday	Start Time	End Time	Department	Task	Employee	Date	Remove
Monday	07:00:00	18:00:00	Kitchen	Barista	Arvid Arbetare		
Monday	09:00:00	17:00:00	Office	Manager	Jens Klemets		
Monday	08:00:00	16:00:00	Kitchen	Dish washer	Patrik Nylund		
Tuesday	08:00:00	16:00:00	Kitchen	Barista	Arvid Arbetare		
Tuesday	07:00:00	18:00:00	Kitchen	Dish washer	Patrik Nylund		
Tuesday	09:00:00	17:00:00	Office	Manager	Jens Klemets		
Wednesday	09:00:00	17:00:00	Office	Manager	Jens Klemets		
Wednesday	08:00:00	15:00:00	Kitchen	Dish washer	Patrik Nylund		
Wednesday	07:00:00	18:00:00	Kitchen	Barista	Arvid Arbetare		
Thursday	06:00:00	15:00:00	Kitchen	Barista	Arvid Arbetare		
Thursday	09:00:00	17:00:00	Office	Manager	Jens Klemets		
Friday	09:00:00	17:00:00	Kitchen	Barista	Arvid Arbetare		
Friday	08:00:00	16:00:00	Kitchen	Dish washer	Patrik Nylund		
Friday	09:00:00	17:00:00	Office	Manager	Jens Klemets		
Monday	12:00:00	15:00:00	Counter	Rush hour help	Emilia Extra	2013-03-04	<a href="#">Remove</a>
Friday	12:00:00	15:00:00	Counter	Rush hour help	Emilia Extra	2013-03-08	<a href="#">Remove</a>

Figur 27. Vyn för all planerad tid

## 4.4 Anställdas vy

De anställdas vy har som administratörns vyn en Dashboard. Dashboarden är inte konfigurerad i alfa versionen. Man har två vyer till som anställd, en för skiften och en för att ändra sin information.

Vyn för att se den planerade arbetstiden visar en tabell med information om skiften man har bokat. Det finns en länk man kan klicka på för att komma till skiftesbokningsvyn. Skiftesbokningen har två fält, ett som visar en lista på vilka avdelningars skiften man kan boka och ett där man skall välja datum man vill arbeta. När man klickar på datumet man vill arbeta visas en popup med alla skiften som är tillgängliga att boka ifall sådana finns.

Workshift Alpha						
Dashboard <b>Planned time</b> Your information Logout						
Planned work						
Book shift						
Department	Day	Start time	End time	Task	Date	
Counter	1	12:00:00	15:00:00	Rush hour help	2013-03-04	
Counter	5	12:00:00	15:00:00	Rush hour help	2013-03-08	

Figur 28. Vy med bokade extra skiften

Figur 29. Vyn för att boka ett extra skifte

En anställd kan också behöva ha till gång till att ändra på informationen om sig själv. Man har därför skapat en vy där den anställda kan ändra den egna informationen. Den anställda har dock inte tillgång att sätta till sig själv i en avdelning eftersom man anser att det är bara administratörn som skall ha rätt till det.

Figur 30. Vyn för att editera anställds information



## 5 DISKUSSION

Ramverk är ett mycket effektivt sätt att utveckla en webbapplikation. Man har med väl-skriven dokumentation fått insikt i hur man skall gå tillväga när man använder sig av ett ramverk. Det som var svårast i hela projektet var att tänka sig hur man vill visa informationen i databasen åt användaren. Projektet har gett mig en helt ny syn på applikations-utveckling och har varit en mycket givande process.

CodeIgniter har ett mycket lättanvänt bibliotek vilket har påverkat utvecklingen av applikationen positivt. Skulle jag ha börjat utveckla utan ett ramverk och skrivit all PHP koden för hand skulle utvecklingstiden ökat avsevärt. Jag skulle också ha fått mycket mera buggar i applikationen och haft det svårare att göra applikationen säker. CodeIgniter uppdateras med jämna mellanrum för att öka effektiviteten och gör det mera säkert.

Jag har sett mycket utvecklingsmöjligheter i applikationen. För det första vill jag få bort tabellkänslan ur den färdiga applikationen. En kalendervy kan skapas med hjälp av CodeIgniters kalenderbibliotek. Applikationen kan också göras säkrare genom att göra ännu effektivare kontroller av information som skickas till databasen. Applikationen kunde också behöva en funktion för att exportera arbetstiderna till de populära digitala kalendrar som finns.

## KÄLLOR

Oracle Corporation 2013 | Market Share, [www] Tillgänglig:

<http://www.mysql.com/why-mysql/marketshare/> Hämtad 22.1.2013

Ellislab, Inc 2012 | User Guide, [www] Tillgänglig: <http://ellislab.com/codeigniter/user-guide/overview/mvc.html> Hämtad 4.2.2013

Oracle Corporation 2013 | MySQL Customers by Industry, [www] Tillgänglig:

<http://www.mysql.com/customers/industry/> Hämtad 23.1.2013

Oracle Corporation 2013 | Chapter 7. Data Modeling, [www] Tillgänglig:

<http://dev.mysql.com/doc/workbench/en/wb-data-modeling.html> Hämtad 23.1.2012

Abeyasinghe, Samisa. 2009, PHP Team Development. 1 uppl., USA: Packt Publishing, 169s., ISBN 978-1-847195-06-7

Liu, Chamond. 1999, SmallTalk, Objects, and Design. Nytryck. USA: ToExcel, 289s., ISBN 1-58348-490-6

Benedetti, Ryan & Cranley Ronan. 2011, Head First jQuery. 1 uppl., USA: O'Reilley Media 500s., ISBN 978-1-449-39321-2

Mozilla Developer Network 2013 | About JavaScript, [www] Tillgänglig:

[https://developer.mozilla.org/en-US/docs/JavaScript/About\\_JavaScript](https://developer.mozilla.org/en-US/docs/JavaScript/About_JavaScript) Hämtad 26.1.2013

Q-Success 2013 | Usage of JavaScript libraries for websites, [www] Tillgänglig:

[http://w3techs.com/technologies/overview/javascript\\_library/all](http://w3techs.com/technologies/overview/javascript_library/all) Hämtad 26.1.2013

Resig, John 2009 | jQuery 1.3 and the jQuery Foundation, [www] Tillgänglig:  
<http://blog.jquery.com/2009/01/14/jquery-13-and-the-jquery-foundation/> Hämtad  
27.1.2013

Sizzle 2013 | Selector Features, [www] Tillgänglig: <http://sizzlejs.com/> Hämtad  
27.1.2013

Modernizr 2013 | Documentation, [www] Tillgänglig: <http://modernizr.com/docs/> Hämtad 27.1.2013

W3Schools.com 2013 | Higher Screen Resolutions, [www] Tillgänglig:  
[http://www.w3schools.com/browsers/browsers\\_resolution\\_higher.asp](http://www.w3schools.com/browsers/browsers_resolution_higher.asp) Hämtad 2.2.2013